BEES Workshop, Dec 4-6, 2000
Jet Propulsion Laboratory

Stochastic Optimization using Genetics-Inspired Search Techniques

Barbara Engelhardt, Jet Propulsion Laboratory
Steve Chien, Jet Propulsion Laboratory
{firstname.lastname}@jpl.nasa.gov

Abstract

Proposed missions to explore comets and moons will encounter environments that are hostile and unpredictable. Any successful explorer must be able to adapt to a wide range of possible operating solutions to survive. Constructing special-purpose behaviors requires information about the environment, which is not available *a priori* for these missions. Instead, the explorer must use a flexible problem-solver with a significant capacity to adapt its behavior. More specifically, the explorer must use stochastic optimization techniques to continually adapt its behavior while limiting the cost for exploration. With Adaptive Problem Solving, we take a biologically-inspired approach to both stochastic optimization and search in order to enable a spacecraft to adapt its environment-specific behavior in-situ.

Introduction

Optimizing the decision-making policy of an autonomous agent is a difficult problem because of the large number of possible behaviors and behavior combinations; the problem only becomes more difficult when the utility feedback function is a distribution instead of a single value. Stochastic optimization is defined for an adaptive problem solver as the problem of optimizing the expected utility of a hypothesis for a certain domain. Our adaptive problem solver optimizes the decision-making policy used by a planning system (for this work, the Automated Scheduling and Planning Environment, or ASPEN [Chien 2000]), which is autonomously controlling a spacecraft. The planner searches through the space of all possible plans to find a single plan that accomplishes the mission goals while satisfying all of the resource and temporal constraints of the model in the current know state of the world. The control strategy to be optimized is expressed as a vector of weights on heuristic functions used to control the search. Adapting the decision-making policy will impact the generated plans and thus, at a high level, the overall behavior of the spacecraft.

There are two parts to adaptation by stochastic optimization: evaluating hypothesis decision-making strategies and generating the set of candidate strategies. The evaluation component generates a sufficient number of samples for each of the hypothesis behaviors to satisfy a given decision criterion in order to rank the hypothesis behaviors within given statistical bounds [Chien 1995]. The generation method uses the ranking from the evaluation step to generate the subsequent set of hypothesis behaviors to be evaluated. The cycle stops when the highest estimated utility has reached quiescence or a time limit has been reached.

Optimization is performed in biological systems through natural selection and reproduction. In adaptive problem solving, the two parts of the process are combined in an evolutionary method, such that the process mimics evolution, or adaptation through generations. Standard computational optimization focuses on a single parent and a single offspring, which is evaluated and then accepted or rejected as the next parent. Adaptive problem solving uses a ranked set of parents and creates a generation of children, which are conceived from the higher-ranking parents. Furthermore, the generation methods are based on genetic recombination, including crossover, mutation, and reproduction, to imitate genetic search.

The number of possible decision-making strategies to select over is enormous. Local search techniques could be used to perform optimization within this search space effectively. But local search techniques will not work well in the search space unless gradient information is used, the space is smooth, or at a very minimum the space is continuous. In addition, the landscape of the search space is defined by the neighborhood for the step function. This means two vector hypotheses that are one neighborhood step away from each other using a single search function could be many steps from each other using another. So to search this space effectively, we must find a search technique that creates a smooth, easily searchable space.

In this paper we will describe the evolutionary process of adaptive problem solving. We will evaluate three different search techniques, random search, local beam search, and genetics-inspired search, with a special emphasis on the genetics-inspired search. The difficulty of searching a landscape is tightly coupled to the landscape properties of ruggedness, distribution of local optima, number of local optima, and topology of attraction basins. These properties will be quantified for the three search techniques [Stadler 2000]. Furthermore, we compare the actual results of applying the three different search algorithms. We will show how effective genetics-inspired techniques are for a few large optimization domains compared to beam search and random search.

Europa Submersible Scenario

Consider NASA's planned mission to Jupiter's moon, Europa, which will send a robot to the surface of the moon Europa in order to analyze the ice surface and hypothesized ocean underneath. Once the submersible has dug through the ice, it might have the scientific goals of moving to the next target 40% of the time allotted to achieving goals, and imaging or performing scientific experiments 60% of that time. All of the images must be uplinked to the orbiter, and an uplink should happen once every orbit in order to ensure that the image buffer capacity will not be exceeded. Many scenarios would force these percentages and plans to be adapted. One scenario might be that movement in the ocean is much more resource-intensive than expected, in which case moving 40% of the time would not allow the submersible to make as much progress as previously expected, so perhaps it might choose to increase this amount. Another scenario might be that the ice layer is much thicker than previously imagined and uplinking the on-board data to the orbiter requires special positioning of the submersible. In this case, it might be better to uplink once every few orbits to let the submersible have more time to collect experiments before it gets into the uplink position. This creates the need to increase the percentage of time devoted to moving, since the uplink position will require movement to and from the current position of the submersible. The increase in the percentage of time devoted to movement will cause less science to be taken, which is synergistic with the adaptation for fewer uplinks.

Failure to adapt to these situations could cost the submersible the mission, by depleting resources too rapidly, not accomplishing mission objectives, or wearing out equipment. Not all possible situations can be enumerated before the mission; instead an adaptive problem solver checks the current control strategy's performance in the given environment and responds to changes by adapting the decision-making strategy, independent of the cause of the change. An adaptive problem solver would continually adapt the control strategy if it found the current strategy non-optimal.

In actual operations, the submersible would have a planning system controlling its decision-making processes using a chosen behavior. The preliminary behavior would be responsible for preserving the safety of the submersible while it learned about its new environment. In a separate process, an alternate planner and simulator would be adapting the current behavior to include movement and science goals, while the simulator receives constant updates from the submersible as to the current state of the environment. When a decision-making strategy is learned in the separate process that preserved the safety of the submersible and accomplished mission goals, it could be swapped into the primary planning system and used operationally. If this decision-making strategy began to degrade in performance, the primary planning system would request a new strategy, and update the simulator with possible environmental changes or spacecraft degradations. Thus adaptation relies on environmental feedback to improve the current decision-making strategy on-board a spacecraft, while maintaining the safety of the spacecraft.

Planning Domain

We investigate stochastic optimization in the context of learning control strategies for the ASPEN planner [Chien 2000]. ASPEN uses heuristics to facilitate the iterative search for a feasible plan. During each search step, a planner confronts a series of decisions such as which schedule conflict to repair or the action to take to repair it. The planner resolves these choices by stochastically applying the heuristics, based on weights for each choice point heuristic, during iterative repair [Zweben 1994]. Thus the weights define the control strategy of the planner, which impacts the expected utility of the resulting plans.

Specifically, in our setup, a strategy hypothesis is a vector with a weight for each heuristic function and a weight of 0 for a heuristic not in use. The utility of a hypothesis can be determined by running the planner using the control strategy hypothesis on a certain problem instance and scoring the resulting plan. A problem generator for each domain provides a stochastic set of problem instances to enhance the robustness of the expected solution for the entire planning domain.

In our ASPEN planning system, there are twelve choice points in the repair search space. Higher level choice points include choosing the conflict to resolve and choosing the resolution method, such as preferring open constraints before violated constraints, or preferring to add activities over moving them. Once a resolution method is selected, further choice points influence applications of the choice point such as where to place a newly created activity and how to instantiate its parameters. For each choice point, there are many heuristics that might be used. The hypothesis vector is the list of relative weight that is given to each heuristic for that choice point. Since the planner is stochastic, the choice of heuristics that are used at each step is randomized, so multiple runs even for the same problem instance may yield a range of solutions (plans) and hence a distribution of utilities.

Domains

The repair heuristics were developed for individual domain search requirements from ASPEN applications [Chien 2000]. There are also domain-specific heuristics, which reference particular features of a domain in order to affect the search. For each domain, the human expert strategy hypotheses were derived independently from (and prior to) our study by manual experimentation and domain analysis.

We examine two different spacecraft domains, which satisfy the normality assumption of the evaluation method. The first domain, Earth Orbiter-1 (EO-1), is an earth imaging satellite. The domain consists of managing spacecraft operations constraints (power, thermal, pointing, buffers, telecommunications, etc.) and science goals (imaging targets and calibrating instruments with observation parameters). Each problem instance is used to create a two-day operations plan: a typical weather and instrument pattern, observation goals (between 3 and 16), and a number of satellite passes (between 50 and 175). EO-1 plans prefer more calibrations and observations, earlier start times for the observations, fewer solar array and aperture manipulations, lower maximum value over the entire schedule horizon for the solar array usage, and higher levels of propellant [Sherwood 1998].

The Comet Nucleus Sample Return (CNSR) domain models landed operations of a spacecraft designed to land on a comet and return a sample to earth. Resources include power, battery, communications, RAM, communications relay in-view, drill, and ovens. Science includes mining and analyzing a sample from the comet, and imaging. The problem generator includes between 1 and 11 mining activities and between 1 and 24 imaging activities at random start times. The scoring functions for the CNSR domain includes preferences for more imaging activities, more mining activities, more battery charge over the entire horizon, fewer drill movements, and fewer uplink activities.

Adaptive Problem Solving

We adopt an iterative approach to statistical optimization, as defined by the following algorithm:

```
Iterative Stochastic Optimization
For a set of $n$ hypotheses $H$
While (time not exceeded || quiescence not reached)
    Step 1: Using a decision criterion, select the best $h_i$ from $H$ by sampling in the domain
    Step 2: Using $h_i$, generate $H'$ using a chosen neighborhood function; $H := H'$
Return current $h_i$
```

As in evolution, there are two algorithmic parts of this optimization method: ranking the best hypotheses from a given set of hypotheses using hypothesis evaluation (natural selection), and generating a new set of hypotheses using a neighborhood function and the current rankings using strategy optimization (survivor breeding). While the focus of this paper is on the later, we will quickly discuss the hypothesis evaluation methods used in this adaptive problem solver.

Hypothesis Evaluation

Hypothesis Evaluation is the key process in stochastic optimization of determining which of the possible decision-making strategy hypotheses outperforms the others for a specific set of problem instances in the domain. Because the utility *U(h)* cannot be found directly, sampling is used to estimate the expected performance for each decision-making strategy hypothesis. A *statistical decision criterion*, which may take several forms, determines when enough samples have been taken to be reasonably sure of the relative standing of strategy hypothesis *h*, within some user-defined confidence or error bounds. In adaptive systems, it is often necessary to try to reduce the number of samples taken, as testing different decision-making strategies in-situ can be very expensive. Rational forms of the decision criteria optimize the resource usage of the selection algorithm by allocating more samples to hypotheses with less certain utility estimates, thus enabling a confident decision to be made while reducing sampling cost.

In the actual evaluation, initial samples are taken to generate a starting expected utility and estimated variance and select the top control strategy hypothesis thus far. Then the algorithm checks to see whether each pairwise comparison between h_{sel} and the other hypotheses is statistically correct using a decision criterion. If the comparison satisfies the criterion, no more samples are taken to compare the two; if the comparison fails to satisfy the criterion, additional samples are taken until the criterion is satisfied for all of the comparisons. The idea of selection by pairwise comparisons is similar to a common implementation of tournament ranking schemes in genetic programming.

There are many possible decision criteria that can be plugged in to the evaluation function. Some simplify the problem by making the assumption that the stochastic data is normally distributed, while others do not. Our current work has focused on the Probably Approximately Correct (PAC) requirement [Valiant 1984], to determine when the utility of one hypothesis is superior to another based on pairwise comparisons. This decision criterion assumes normally distributed data. With the PAC decision criterion, an algorithm makes decisions with a given confidence (probability 1-δ, for small δ) to select a good hypothesis (within error ε of the best hypothesis). Because any specific decision either satisfies or does not satisfy this requirement, the PAC criterion holds that over a large number of decisions that the accuracy rate must meet 1-δ. For data that is not normally distributed, *distribution-free* exponential bounds (e.g., due to Chernov [Hagerup & Rub, 1990], Hoeffding [Hoeffding, 1963], and Bernstein [Bernstein, 1946]) may be substituted for the exact normal-theory probabilities above. We can use such exponential inequalities to get accurate bounds on the probability of incorrect selection, retaining a probabilistic guarantee, while avoiding the normality assumption.

Strategy Optimization

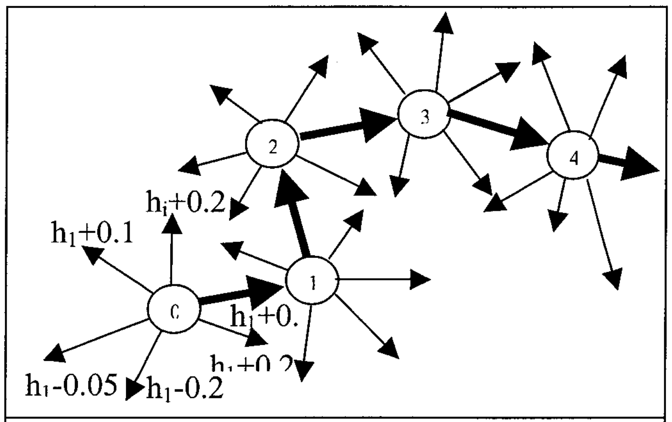


Figure 1: Strategy optimization with a neighborhood search step

For strategy optimization, we use Memetic Algorithms [Memetic 2000, Merz & Freisleben 1999], which incorporate both local search methods and evolutionary computation approaches. The search methods that we are currently using include a local beam search algorithm, a genetic algorithm, and random sampling. Search is conducted in the following manner: we begin with an initial control strategy h_0, or in the case of genetic search a set of ranked control strategies, and a candidate step function for each of the search algorithms. The candidate function generates a set of next hypotheses. The hypothesis ranking system ranks the new set of hypotheses and choses the best one to become h_1, or in the case of the genetic algorithm, chooses a subset of highest ranking hypotheses for the subsequent generation step. The search algorithm generates another set of hypotheses based on using the step function with h_1, and this process continues until quiessence or a time limit is exceeded.

The three current search functions used are genetics-inspired search, local beam search, and random search. Genetic search similarly takes the top n ranking hypotheses, and uses them as the parents to the offspring of the next generation. The neighborhood functions are one of the three basic genetic operators: reproduction, crossover, and mutation. For reproduction, a single parent is duplicated exactly in the next generation based on its expected utility. For crossover, two parent strategy hypotheses are chosen based on their estimated utility to split and recombine at some point *k* in their vector to form two offspring. For mutation, a single element of the offspring vector is mutated with some probability.

Local beam search takes the top *b* (where *b* is the size of the beam) ranked strategy hypotheses, and builds the next candidate generation using those hypotheses, plus an even number of neighborhood steps form each of those hypotheses. A neighborhood step is defined as a mutation of each element of the control strategy hypothesis vector up to a certain total percentage. This is equivalent to reproduction and mutation, although the mutation in local search bounds the amount that the vector element can be mutated.

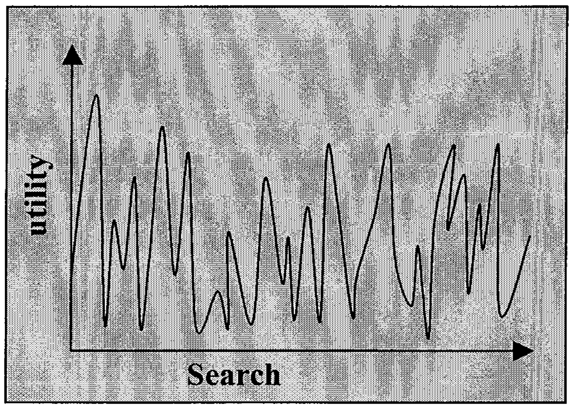
Random search does not rely on the previous generation; instead it creates a new generation from random, independent points in the search space.

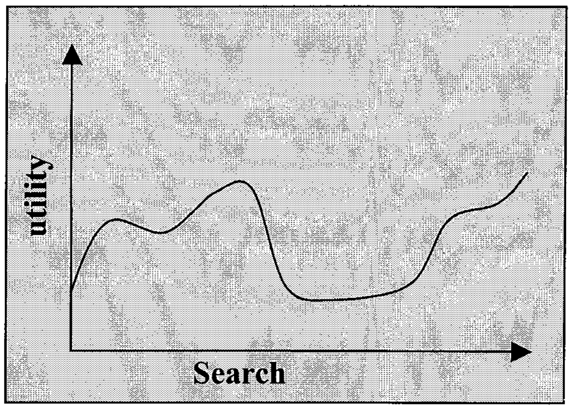
Results

We investigated two different aspects of the performance of each candidate generation method: ruggedness and directed search. Each performance element reinforced the benefits of using genetic algorithms on a domain of this type, specifically an unsearchably large domain with a large number of local maxima.

Ruggedness

Ruggedness is a measure of the correlation between adjacent (or neighborhood) points for a given neighborhood function. In order for local search to be effective, adjacent points should be correlated in terms of their expected utility. Without correlation between neighborhood points, random search would be just as effective as local search on that particular domain (see figure 2a). But if neighborhood points are correlated, then local search can use collected samples to learn and search gradients, or positive trends, in the search space in order to reach maxima.





Figures 2a,b: Graphs show a rough, less continuous space versus a smooth, more continuous space. The efficacy of local search algorithms is correlated with characteristics (including smoothness) of the search landscape they produce.

One simple way to test the ruggedness of a particular neighborhood algorithm in a search space is to measure the average change in expected utility of a single step in the search space. For random search, adjacent points can be any two points in the search space. For genetic search, the difference in utility between two adjacent points can be measured by calculating the expected utility of the parents relative to a single offspring. For local search, two adjacent points can be a starting vector and its adapted vector.

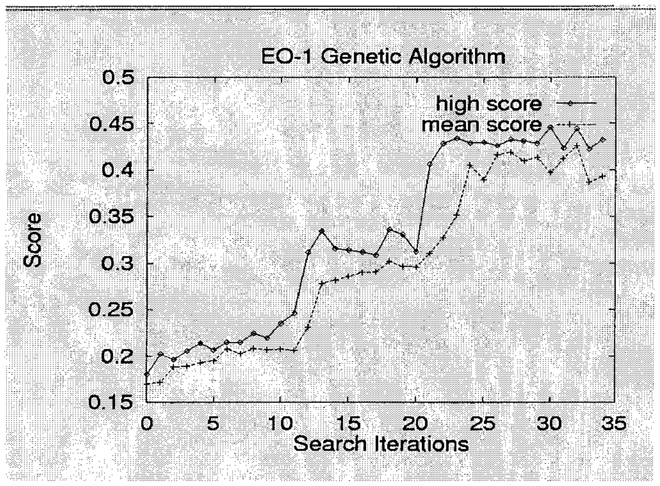
Ruggedness does not entirely determine the smoothness of the search space. In the example of a rugged graph above, consider reducing the height by two-thirds. It would appear that the measure of

Domain	Random Search		Local Beam Search		Genetic Search	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
Comet Lander	0.0435	0.0293	0.0086	0.0066	0.0134	0.0093
EO-1	0.0442	0.0466	0.0114	0.0331	0.0145	0.0244

Table 1: Smoothness property of each search algorithm for both domains. The mean shows the average change in expected utility between two steps of the search, which is a measure of the smoothness of the step function.

smoothness for the rugged, unsearchable graph would be similar to the smoothness of the example of the smooth, easily searchable graph. Since the rugged graph is still unsearchable, even though the neighborhood points are relatively close to one another, another measure of ruggedness must be the number of local maxima that is found for particular search methods. This will tell us the approximate order of how many peaks the landscape has, and give us a more accurate estimate of the smoothness of the landscape. Tests on the local maxima are still being performed.

Overall Performance



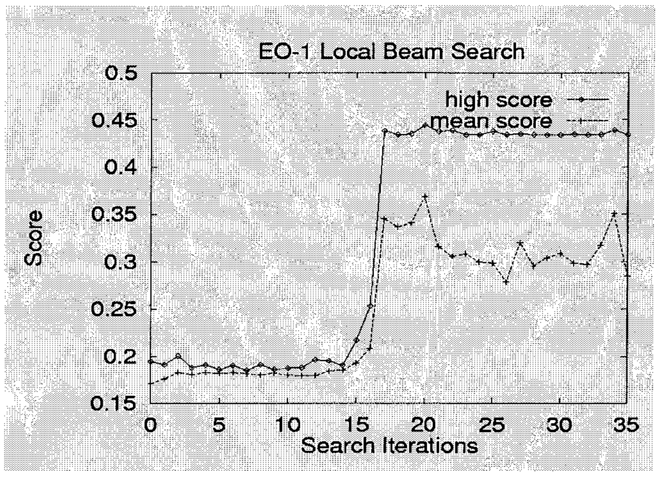


Figure 3a,b: Preliminary results of the improvements made by genetics-inspired search algorithm on a planning domain. Each point represents the estimated value of the high and mean scores for each generation of hypotheses, which is why the maximum can be seen to decrease occasionally.

Given the same starting points, genetic search and local search made significant improvements, and performed very well compared with the best hypothesis found in an equivalent number of random samples. There were two features of the genetic search that enable a more robust search for any given starting hypothesis (see figure 3a). First, the improvements have a constant upwards slope until the plateau. Unlike the local search method, which made most of the improvement in a single search iteration (see figure 3b), genetic search had relatively consistent improvements in the mean and high scores for each generation of hypotheses throughout the entire optimization run. This consistent improvement makes genetic search a more useful anytime algorithm, and when CPU resources are limited, a small number of iterations will most likely be more effective than a few steps of local search optimization.

Second, the mean of each generation is much closer to the high score, meaning that the average of the entire generation as a whole has improved. This illustrates that the landscape created using genetic search is searchable in practice, as siblings from high scoring parents are much more tightly clustered around a similarly high score than the sibling offspring from high scoring parents using local search.

These preliminary results show the promise of using a genetic algorithm for an anytime adaptation algorithm. Further experiments are being produced to confirm and expand on this hypothesis.

Related Work

Evaluating decision-making strategies is a growing research topic, although many of the methods employed in the related work do not rely on any evolutionary-based computational strategies. Horvitz originally described a method for evaluating algorithms based on a cost versus quality tradeoff [Horvitz 1988]. Russell et al. used dynamic programming to rationally select among a set of control strategies by estimating utility [Russell et al. 1993]. The MULTI-TAC system considers all k-wise combinations of heuristics for solving a CSP in its evaluation which also avoids problems with local maxima, but at a large expense to the search [Minton 1996]. Previous articles describing adaptive problem solving have discussed general methods for transforming a standard problem solver into an adaptive one [Gratch & DeJong 1992, 1996], illustrated the application of adaptive problem solving to real world scheduling problems [Gratch & DeJong 1996], and showed how adaptive problem solving can be cast as a resource allocation problem [Chien 1999]. We have expanded on these topics by evaluating methods for generating hypotheses from biologically-inspired algorithms, and using adaptive problem solving to evaluate those candidate hypotheses based on an efficient estimate of their utility and cost, considered separately in the ranking process.

Future Work and Conclusions

Genetics-inspired search can be improved in a number of ways. We have already begun using simulated annealing cooling functions to reduce the amount of mutation, the probability of choosing an inferior parent for reproduction or crossover, or scaling the utility function [Goldberg 1989]. We plan on using distance correlation functions and covariance analyses to determine the relative dependence of the elements in the hypothesis vector, so that we can place them in the vector such that highly correlated choice point heuristic weights are closer, and non-correlated choice point heuristic weights are far apart on the vector. The relative distance compared with correlation impacts the search by decreasing the likelihood that correlated choice points will be split during a crossover operation [Goldberg 1989].

We will continue to perform landscape analysis to measure and characterize the epistatis of the search space, the number of local maxima, and the average distance between local maxima (to test for the "big valley" hypothesis). We are also looking into meta-level learning techniques such as landscape mapping functions and multi-restart techniques in order to better evaluate search strategy techniques.

Bibliography

1. Bernstein, S.N. (1946). *The Theory of Probabilities*, Gastehizdat Publishing House.

2. Chien, S., Rabideau, G., Knight, R., Sherwood, R., Engelhardt, B., Mutz, D., Estlin, T., Smith, B., Fisher, F., Barrett, T., Stebbins, G., Tran, D. 2000. "ASPEN – Automating Space Mission Operations using Automated Planning and Scheduling." *SpaceOps 2000,* Toulouse, France.

3. Chien, S., Stechert, A., Mutz, D. 1999. "Efficient Heuristic Hypothesis Ranking." *Journal of Artificial Intelligence Research* Vol 10: 375-397.

4. Chien, S., Gratch, J., Burl, M. 1995. "On the Efficient Allocation of Resources for Hypothesis Evaluation: A Statistical Approach." In *Proceedings of the IEEE Transactions on Pattern Analysis and Machine Intelligence* 17(7), p. 652-665.

5. Goldberg, D., *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Publishing Company, USA. January1989.

6. Gratch, J; DeJong, G. 1996. A Decision-Theoretic Approach to Solving the EBL Utility Problem. In *Artificial Intelligence.*

7. Gratch, J. DeJong, G. 1992. COMPOSER: A Probabilistic Solution to the Utility Problem in Speed-up Learning. In *Proceedings of the Tenth National Conference on Artificial Intelligence.*

8. Hagerup, T. and Rub, C. (1990). A Guided Tour of Chernov Bounds, Information Processing Letters 33: 305-308.

9. Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables, Journal of the American Statistical Association 58(301):13-30

10. Horvitz, E. 1988. Reasoning under Varying and Uncertain Resource Constraints. *Proceedings of the Seventh National Conference on Artificial Intelligence,* 111-116.

11. Minton, S. 1996. Automatically Configuring Constraint Satisfaction Programs: A Case Study. In *Constraints* 1:1(7-43).

12. Russell, S., Subramanian, D., and Parr, R. 1993. Provably Bounded Optimal Agents. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence.*

13. Sherwood, R., Govindjee, A. Yan, D., Rabideau, G., Chien, S., Fukunaga, A. 1998. Using ASPEN to automate EO-1 Activity Planning. In *Proceedings of the 1998 IEEE Aerospace Conference.*

14. Stadler, P., "The Structure of Fitness Landscapes," *Proc. of International Workshop on Biological Evolution and Statistical Physics*, Max Planck Institute for Complex Systems Physics, May 2000.

15. Valiant, L. "A Theory of the Learnable," *Communications of the ACM*, 1984.

16. Zweben, M. Daun, B., Davis, E., and Deale, M. 1994. Scheduling and Rescheduling with Iterative Repair. In Zweben, M., and Fox, M., eds., *Intelligent Scheduling.* San Francisco, CA: Morgan Kaufmann. 241-256.